Politechnika Białostocka
Wydział Elektryczny
Katedra Elektrotechniki, Energoelektroniki i Elektroenergetyki

Instrukcja
do pracowni specjalistycznej z przedmiotu

Programowanie w języku C

Kod przedmiotu: **TZ1E1004BK**

(studia niestacjonarne)

JĘZYK C - ŁAŃCUCHY ZNAKÓW

Numer ćwiczenia

PROGwC_07

Autor:
dr inż. Jarosław Forenc

Białystok 2025

Spis treści

1. Opis stanowiska 3

1.1. Stosowana aparatura 3

1.2. Oprogramowanie 3

2. Wiadomości teoretyczne 3

2.1. Deklaracja łańcucha znaków 3

2.2. Inicjalizacja łańcucha znaków 4

2.3. Stała znakowa 6

2.4. Funkcje do wyprowadzania i wprowadzania pojedynczych znaków oraz łańcuchów znaków 7

2.5. Funkcje z pliku nagłówkowego string.h 13

3. Przebieg ćwiczenia 15

4. Literatura 17

5. Pytania kontrolne 17

6. Wymagania BHP 17

Materiały dydaktyczne przeznaczone dla studentów Wydziału Elektrycznego PB.

1. Opis stanowiska

1.1. Stosowana aparatura

Podczas zajęć wykorzystywany jest komputer klasy PC z systemem operacyjnym Microsoft Windows 10/11.

1.2. Oprogramowanie

Na komputerach zainstalowane jest środowisko programistyczne Code::Blocks.

2. Wiadomości teoretyczne

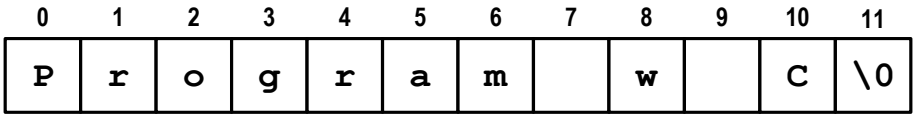
2.1. Deklaracja łańcucha znaków

Łańcuch znaków (ciąg znaków, napis, stała napisowa, stała łańcuchowa, literał łańcuchowy, C-string) jest to ciąg złożony z zera lub większej liczby znaków zawartych między znakami cudzysłowu, np.

```
"Program w C"
```

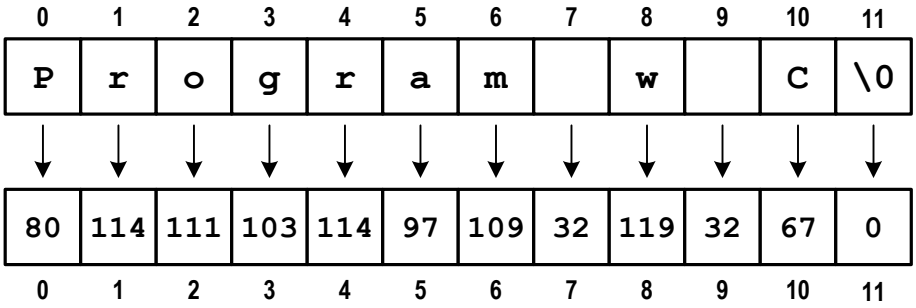
Znaki cudzysłowu nie są częścią napisu, służą jedynie do określenia jego granic.

Łańcuchy znaków przechowywane są w postaci tablicy, której elementami są pojedyncze znaki (tablica elementów typu **char**). Ostatnim elementem tablicy jest znak o kodzie **0** (stała liczbowa **0** lub stała znakowa **'\0'**), oznaczający koniec napisu (Rys. 1). Większość funkcji wykonujących operacje na łańcuchach znaków dodaje ten znak automatycznie.



Rys. 1. Tablica przechowująca tekst „Program w C”

W rzeczywistości w tablicy zamiast znaków przechowywane są odpowiadające im kody ASCII czyli liczby (Rys. 2).



Rys. 2. Reprezentacja znaków w tablicy

Deklaracja zmiennej mogącej przechowywać łańcuch znaków jest podobna do deklaracji zwykłej tablicy:

```
char nazwa_zmiennej[rozmiar];
```

Rozmiar fizycznej pamięci przeznaczonej na napis musi być o jeden większy niż liczba znaków zawartych między znakami cudzysłowu. Deklaracja tablicy, w której można przechowywać napisy o maksymalnej długości do 20 znaków ma postać:

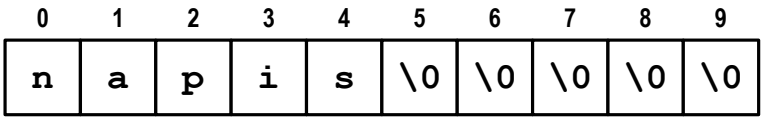
```
char str[21];
```

2.2. Inicjalizacja łańcucha znaków

Deklarując łańcuch znaków możemy nadać mu wartość początkową, np.

```
char str[10] = "napis";
```

Pozostałym elementom tablicy automatycznie przypisywana jest stała liczbowa **0** czyli stała znakowa **'\0'** (Rys. 3).



Rys. 3. Tablica znaków po inicjalizacji

Inicjalizując łańcuch znaków można również podać pojedyncze znaki umieszczone w apostrofach, np.

```
char str[10] = {'n','a','p','i','s'};
```

lub odpowiadające im kody ASCII:

```
char str[10] = {110,97,112,105,115};
```

Deklarując tablicę można nie określać jej długości, kompilator przydzieli wtedy automatycznie odpowiedni rozmiar pamięci (uwzględniając ostatni znak **'\0'**):

```
char *str = "napis";
```

lub

```
char str[] = "napis";
```

W powyższy sposób można nadawać wartość łańcuchowi znaków tylko przy jego deklaracji. Zatem błędne jest poniższe przypisanie:

```
char str[10];
str = "napis";
```

Poprawne zapisanie (skopiowanie) tekstu to tablicy wymaga wywołania funkcji **strcpy()** z pliku nagłówkowego **string.h**:

```
char str[10];
strcpy(str,"napis");
```

Funkcje znajdujące się w tym pliku nagłówkowym zostały opisane w dalszej części instrukcji (Rozdział 0).

2.3. Stała znakowa

Stała znakowa jest to liczba całkowita. Taką stałą tworzy jeden znak ujęty w apostrofy, np. **'x'**. Wartością stałej znakowej jest wartość kodu ASCII (Tabela 1).

Tabela 1. Wybrane kody ASCII

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
32	20	Space	56	38	8	80	50	P	104	68	h
33	21	!	57	39	9	81	51	Q	105	69	i
34	22	″	58	3A	:	82	52	R	106	6A	j
35	23	#	59	3B	;	83	53	S	107	6B	k
36	24	$	60	3C	<	84	54	T	108	6C	l
37	25	%	61	3D	=	85	55	U	109	6D	m
38	26	&	62	3E	>	86	56	V	110	6E	n
39	27	`	63	3F	?	87	57	W	111	6F	o
40	28	(	64	40	@	88	58	X	112	70	p
41	29	)	65	41	A	89	59	Y	113	71	q
42	2A	*	66	42	B	90	5A	Z	114	72	r
43	2B	+	67	43	C	91	5B	[	115	73	s
44	2C	,	68	44	D	92	5C	\	116	74	t
45	2D	-	69	45	E	93	5D	]	117	75	u
46	2E	.	70	46	F	94	5E	^	118	76	v
47	2F	/	71	47	G	95	5F	_	119	77	w
48	30	0	72	48	H	96	60	`	120	78	x
49	31	1	73	49	I	97	61	a	121	79	y
50	32	2	74	4A	J	98	62	b	122	7A	z
51	33	3	75	4B	K	99	63	c	123	7B	{
52	34	4	76	4C	L	100	64	d	124	7C	|
53	35	5	77	4D	M	101	65	e	125	7D	}
54	36	6	78	4E	N	102	66	f	126	7E	~
55	37	7	79	4F	O	103	67	g	127	7F	DEL

Pewne znaki niegraficzne mogą być reprezentowane w stałych znakowych przez sekwencje specjalne, które wyglądają jak dwa znaki, ale reprezentują tylko jeden znak. Należą do nich:

`'\n'`	nowy wiersz	`'\\'`	\ (ang. backslash)
`'\t'`	tabulator poziomy	`'\''`	apostrof
`'\v'`	tabulator pionowy	`'\"'`	cudzysłów
`'\a'`	alarm	`'\?'`	znak zapytania

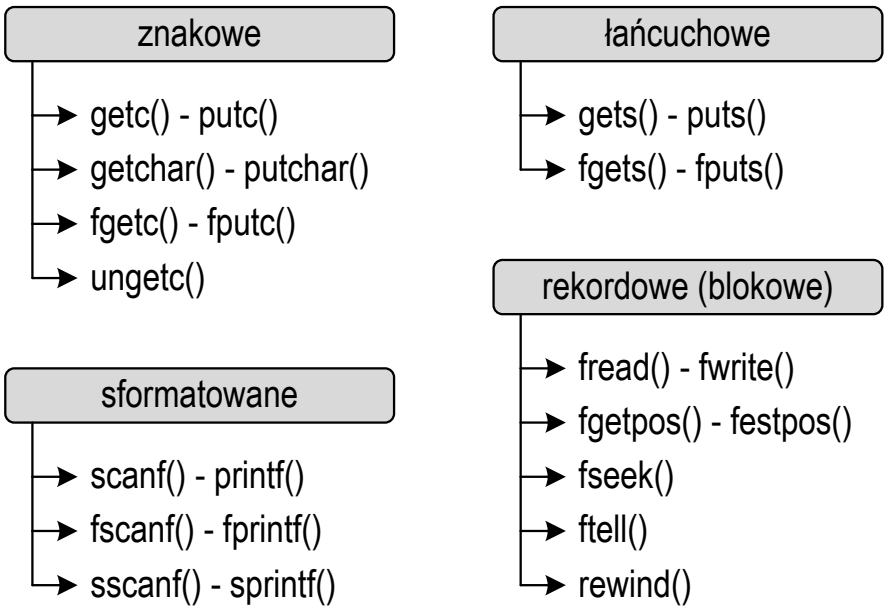
Uwaga: zapis **'A'** oznacza jeden znak, natomiast zapis **"A"** - dwa znaki, gdyż jest to łańcuch znaków kończący się znakiem **'\0'**.

2.4. Funkcje do wyprowadzania i wprowadzania pojedynczych znaków oraz łańcuchów znaków

W języku C standardowe operacje wejścia-wyjścia można podzielić na 4 typy:

- **znakowe** - przetwarzanie danych odbywa się znak po znaku;
- **łańcuchowe** - przetwarzanie danych odbywa się wierszami;
- **sformatowane** - przy przetwarzaniu danych stosowane są specyfikatory formatu;
- **rekordowe (blokowe)** - dane przetwarzane są całymi blokami (rekordami).

Nazwy funkcji wykonujących poszczególne typy operacji przedstawiono na Rys. 4. Zastosowanie tych funkcji wymaga dołączenia pliku nagłówkowego **stdio.h**.



Rys. 4. Typy operacji wejścia-wyjścia w języku C

W przypadku wprowadzania (wczytywania z klawiatury) oraz wyprowadzania (wyświetlania na ekranie) pojedynczych znaków i łańcuchów znaków zazwyczaj stosowane są funkcje:

- znakowe: **getchar()**, **putchar()**;
- łańcuchowe: **gets()**, **puts()**;
- sformatowane: **scanf()**, **printf()**.

Pozostałe funkcje (znakowe, łańcuchowe, sformatowane) przedstawione na Rys. 4 mogą być wykorzystywane do operacji na znakach, ale wymagają dodatkowego argumentu w postaci nazwy standardowego strumienia (**stdin**, **stdout**). Funkcje te zostały opisane w instrukcji **INF28**.

Poniżej opisano funkcje do wyprowadzania pojedynczych znaków oraz łańcuchów znaków.

`putchar()`	Nagłówek: int **putchar(**int **znak);**

- funkcja **putchar()** wyprowadza (wyświetla na ekranie) jeden **znak**;
- jeśli wykonanie zakończyło się poprawnie, to zwraca wypisany **znak**;
- jeśli wystąpił błąd, to zwraca wartość **EOF**;

`puts()`	Nagłówek: int **puts(**const char ***str);**

- funkcja **puts()** wyprowadza (wyświetla na ekranie) łańcuch znaków **str**, zastępując znak **'\0'** znakiem **'\n'** (co oznacza automatyczne przejście do nowego wiersza po wyświetleniu zawartości łańcucha **str**);
- jeśli wykonanie zakończyło się poprawnie, to funkcja **puts()** zwraca ostatni wypisany znak; jeśli wystąpił błąd, to zwraca wartość **EOF**;

`printf()`	Nagłówek: int **printf(**const char ***format, ...);**

- funkcja **printf()** wyprowadza (wyświetla na ekranie) dane zgodnie z podanymi specyfikatorami formatu;

- do wyświetlenia łańcucha znaków stosowany jest specyfikator formatu **%s**, zaś do wyświetlenia pojedynczego znaku - **%c**;
- w specyfikatorze **%s** szerokość określa szerokość pola, zaś precyzja - liczbę pierwszych, wyświetlanych, znaków z łańcucha.

W poniższym programie pokazano różne sposoby wyświetlania pojedynczych znaków oraz łańcuchów znaków.

Program przedstawiający różne sposoby wyświetlenia pojedynczych znaków i łańcuchów znaków (putchar, puts, printf).

```
#include <stdio.h>

int main(void)
{
    char str[16] = "Tekst w tablicy";

    putchar('W');  putchar('i');  putchar('t');
    putchar('a');  putchar('j');  putchar(' ');
    putchar(115);  putchar(119);  putchar(105);
    putchar(0x65); putchar(0x63); putchar(0x69);
    putchar(0145); putchar('\n');

    puts(str);
    puts("Tekst jako argument");

    printf(str);          printf("\n");
    printf("%s",str);     printf("\n");
    printf("%s\n",str);
    printf("%15.10s\n",str);
    printf("Tekst jako argument\n");
    printf("%s\n","Tekst jako argument");

    printf("Znaki: ");
    for (int i=0; i<16; i++) printf("%c_",str[i]);
    printf("\n");

    printf("Znaki: ");
    for (int i=0; i<16; i++)
    {
        putchar(str[i]); putchar('_');
    }
```

```
    printf("\n");

    printf("Kody:  ");
    for (int i=0; i<16; i++) printf("%d ",str[i]);
    printf("\n");

    return 0;
}
```

Wynik uruchomienia programu:

```
Witaj swiecie
Tekst w tablicy
Tekst jako argument
Tekst w tablicy
Tekst w tablicy
Tekst w tablicy
     Tekst w ta
Tekst jako argument
Tekst jako argument
Znaki: T_e_k_s_t_ _w_ _t_a_b_l_i_c_y_ _
Znaki: T_e_k_s_t_ _w_ _t_a_b_l_i_c_y_ _
Kody:  84 101 107 115 116 32 119 32 116 97 98 108 105 99 121 0
```

Argumentem funkcji **putchar()** może być stała znakowa:

```
putchar('W');
```

lub kod ASCII znaku (liczba w jednym z trzech systemów liczbowych: dziesiętnym, ósemkowym, szesnastkowym):

```
putchar(115);     // system dziesietny
putchar(0145);    // system osemkowy
putchar(0x65);    // system szesnastkowy
```

Ponieważ łańcuch znaków jest zwykłą tablicą - można więc odwoływać się do jej pojedynczych elementów. Wyświetlając te elementy przy zastosowaniu specyfikatora formatu **%c** otrzymamy znaki, zaś stosując specyfikator **%d** - odpowiadające im kody ASCII (liczby).

```
printf("Znaki: ");
for (int i=0; i<16; i++) printf("%c_",str[i]);
printf("\n");

printf("Kody:  ");
for (int i=0; i<16; i++) printf("%d ",str[i]);
printf("\n");
```

Poniżej opisano funkcje do wprowadzania (wczytywania z klawiatury) pojedynczych znaków oraz łańcuchów znaków.

`getchar()`	Nagłówek: int **getchar**(void);

- funkcja **getchar()** wprowadza (wczytuje z klawiatury) jeden znak;
- jeśli wykonanie zakończyło się poprawnie, to zwraca przeczytany znak;
- jeśli wystąpił błąd, to zwraca wartość **EOF**;

`gets()`	Nagłówek: char* **gets**(char ***str**);

- funkcja **gets()** pobiera do bufora pamięci wskazywanego przez argument **str** linię znaków z klawiatury;
- wczytywanie jest kończone po napotkaniu znacznika nowej linii **'\n'**, który w buforze pamięci **str** zastępowany jest znakiem końca łańcucha **'\0'**;
- **gets()** umożliwia wczytanie łańcucha zawierającego spacje i tabulatory;
- jeśli wykonanie zakończyło się poprawnie, to zwraca wskazanie do łańcucha **str**; jeśli wystąpił błąd, to zwraca **EOF**;

`scanf()`	Nagłówek: int **scanf**(const char ***format, ...**);

- funkcja **scanf()** wprowadza (wczytuje z klawiatury) dane zgodnie z podanymi specyfikatorami formatu;
- w funkcji **scanf()** do wczytania łańcucha znaków używamy specyfikatora formatu **%s**, zaś do wczytania pojedynczego znaku - **%c**;

- w specyfikatorze **%s** szerokość określa maksymalną liczbę znaków, która zostanie odczytana; zatem funkcja **scanf()** zakończy wczytywanie tekstu po pierwszym białym znaku (spacja, tabulacja, enter) lub w momencie pobrania maksymalnej liczby znaków.

  Wczytanie tekstu funkcją **scanf()** ma następującą postać:

```
char str[15];
scanf("%s",str);
```

Zmienna **str** jest tablicą. Nazwa tablicy jest adresem jej początku w pamięci komputera. Z tego względu przed **str** nie występuje znak **&**.

Funkcja **scanf()** kończy wczytywanie danych po wystąpieniu pierwszego białego znaku (spacja, tabulacja, enter). Jeśli w powyższym przykładzie użytkownik wprowadzi tekst: **"To jest napis"**, to **scanf()** zapamięta tylko pierwszy wyraz: **"To"**. Zapamiętanie całego wiersza tekstu (do naciśnięcia klawisza Enter) wymaga użycia funkcji **gets()**:

```
char str[15];
gets(str);
```

Do wczytania jednego znaku można zastosować funkcję **getchar()**:

```
int znak;
znak = getchar();
```

lub funkcję **scanf()** ze specyfikatorem formatu **%c** (przed zmienną **znak** musi wystąpić operator **&**):

```
int znak;
scanf("%c",&znak);
```

2.5. Funkcje z pliku nagłówkowego string.h

Plik nagłówkowy **string.h** definiuje funkcje do wykonywania operacji na łańcuchach znaków i tablicach. Poniżej przedstawione zostały wybrane funkcje z tego pliku (opis wszystkich funkcji można zaleźć w literaturze - pozycje [7] i [8]).

`strlen()`	Nagłówek: **size_t strlen(**const char ***str);**

- funkcja **strlen()** zwraca długość łańcucha znaków **str**;
- nie bierze pod uwagę znaku **'\0'**;

`strcpy()`	Nagłówek: char ***strcpy(**char ***str1,** const char ***str2);**

- funkcja **strcpy()** kopiuje łańcuch **str2** do łańcucha **str1**;

`strcat()`	Nagłówek: char ***strcat(**char ***str1,** const char ***str2);**

- funkcja **strcat()** dołącza do łańcucha **str1** łańcuch **str2**;

`strchr()`	Nagłówek: char ***strchr(**const char ***str,** int **c);**

- funkcja **strchr()** przeszukuje łańcuch **str** w celu znalezienia pierwszego wystąpienia znaku **c**;
- zwraca wskaźnik do znalezionego znaku lub **NULL**, jeśli znak nie został znaleziony;

`strstr()`	Nagłówek: char ***strstr(**const char ***str1,** const char ***str2);**

- funkcja **strstr()** przeszukuje łańcuch **str1** w celu odnalezienia podłańcucha **str2** zwracając wskazanie do elementu łańcucha **str1**, od którego zaczynają się znaki takie same jak w łańcuchu **str2**, lub **NULL**, gdy **str2** nie występuje w **str1**;

`strcmp()`	Nagłówek: int **strcmp(**const char ***str1,** const char ***str2);**

- funkcja **strcmp()** porównuje łańcuchy **str1** i **str2** z rozróżnianiem wielkości liter;
- zwraca **0**, gdy **str1=str2**, wartość mniejszą od zera, gdy **str1<str2** i wartość większą od zera, gdy **str1>str2**;

`strcmpi()`	Nagłówek: int **strcmpi(**const char ***str1,** const char ***str2);**

- funkcja **strcmpi()** działa jak **strcmp()**, ale bez rozróżniania wielkości liter;

`strlwr()`	Nagłówek: char ***strlwr(**char ***str);**

- funkcja **strlwr()** zamienia w łańcuchu **str** wielkie litery na małe;

`strupr()`	Nagłówek: char ***strupr(**char ***str);**

- funkcja **strupr()** zamienia w łańcuchu **str** litery małe na wielkie;

`strrev()`	Nagłówek: char ***strrev(**char ***str);**

- funkcja **strrev()** odwraca kolejność znaków w łańcuchu **str**;

`strset()`	Nagłówek: char ***strset(**char ***str, int c);**

- funkcja **strset()** wypełnia łańcuch **str** znakiem **c.**

Poniższy program pokazuje przykładowe wykorzystanie wybranych funkcji z pliku nagłówkowego **string.h**.

Przykładowe operacje na łańcuchu znaków.

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    char napis1[] = "Tekst w buforze", napis2[20];
    int  dlugosc;

    printf("napis1: %s\n",napis1);
    dlugosc = strlen(napis1);
    printf("liczba znakow w napis1: %d\n",dlugosc);
    strupr(napis1);
    printf("napis1 (wielkie litery): %s\n",napis1);
    strlwr(napis1);
    printf("napis1 (male litery): %s\n",napis1);
    strcpy(napis2,napis1);
    printf("napis2: %s\n",napis2);
    strrev(napis2);
    printf("napis2 (odwrocony): %s\n",napis2);

    return 0;
}
```

W wyniku uruchomienia programu na ekranie pojawi się:

```
napis1: Tekst w buforze
liczba znakow w napis1: 15
napis1 (wielkie litery): TEKST W BUFORZE
napis1 (male litery): tekst w buforze
napis2: tekst w buforze
napis2 (odwrocony): ezrofub w tsket
```

3. Przebieg ćwiczenia

Na pracowni specjalistycznej należy wykonać wybrane zadania wskazane przez prowadzącego zajęcia. W różnych grupach mogą być wykonywane różne zadania.

1. Napisz program wyświetlający na ekranie kolejne **kody** ASCII (od **30** do **254**) oraz odpowiadające im **znaki**. Przykładowy fragment wydruku:

```
 91 - [  92 - \  93 - ]  94 - ^  95 - _  96 - `  97 - a
 98 - b  99 - c 101 - e 102 - f 103 - g 104 - h 105 - i
106 - j 107 - k 108 - l 109 - m 111 - o 112 - p 113 - q
```

2. Napisz program, który wczytuje z klawiatury jeden wiersz tekstu, a następnie:
   a) wyświetla na ekranie wczytany tekst;
   b) oblicza i wyświetla liczbę znaków znajdujących się w tekście;
   c) oblicza i wyświetla liczbę małych liter oraz liczbę wielkich liter znajdujących się w tekście;
   d) zamienia wszystkie małe litery na wielkie i ponownie wyświetla tekst;
   e) zamienia wszystkie wielkie litery na małe i ponownie wyświetla tekst.

   Uwaga: nie stosuj funkcji z pliku nagłówkowego **string.h**.

3. Napisz program, w którym użytkownik wczytuje z klawiatury trzy liczby całkowite: **h**, **m**, **s** (**h** - godzina, **m** - minuta, **s** - sekunda). Następnie program powinien zapisać do tablicy łańcuch znaków zawierający czas w formacie **hh:mm:ss** (np. **15:05:30**). Wyświetl zawartość tablicy na ekranie (Uwaga: do rozwiązania zadania wykorzystaj funkcję **sprintf()**).

4. Napisz program, który wczytuje tekst z klawiatury (jeden wiersz), a następnie usuwa wszystkie znaki spacji znajdujące się na początku tekstu i na jego końcu, np.

   `"   Ala ma kota   " → "Ala ma kota"`

5. Napisz program, który będzie wczytywał ciąg znaków składający się z zer i jedynek. Następnie program powinien wyznaczyć liczbę serii w ciągu. Seria w ciągu, to podciąg złożony z jednego lub kilku takich samych znaków.

   Przykładowe wywołanie programu:

   ```
   Podaj ciag:   0011101010011
   Liczba serii: 8
   ```

6. Napisz program, który wczytuje jeden wiersz tekstu, a następnie podaje liczbę samogłosek występujących w tym wierszu.

7. Napisz program, który wczytuje jeden wiersz tekstu, a następnie podaje ile w tym wierszu występuje wyrazów.

4. Literatura

[1] Prata S.: Język C. Szkoła programowania. Wydanie VI. Helion, Gliwice, 2016.

[2] Kernighan B.W., Ritchie D.M.: Język ANSI C. Programowanie. Wydanie II. Helion, Gliwice, 2010.

[3] Deitel P.J., Deitel H.: Język C. Solidna wiedza w praktyce. Wydanie VIII. Helion, Gliwice, 2020.

[4] Kochan S.G.: Język C. Kompendium wiedzy. Wydanie IV. Helion, Gliwice, 2015.

[5] King K.N.: Język C. Nowoczesne programowanie. Wydanie II. Helion, Gliwice, 2011.

[6] http://www.cplusplus.com/reference/clibrary - C library - C++ Reference

[7] https://cpp0x.pl/dokumentacja/standard-C/1 - Standard C

[8] https://www.codeblocks.org/ - Code::Blocks

5. Pytania kontrolne

1. Opisz sposób przechowywania łańcuchów znaków (tekstów) w języku C.
2. Przedstaw sposoby inicjalizacji tablicy znaków.
3. Scharakteryzuj funkcje znajdujące się w pliku nagłówkowym **string.h**.

6. Wymagania BHP

Warunkiem przystąpienia do praktycznej realizacji ćwiczenia jest zapoznanie się z instrukcją BHP i instrukcją przeciw pożarową oraz przestrzeganie zasad w nich zawartych.

W trakcie zajęć laboratoryjnych należy przestrzegać następujących zasad.

- Sprawdzić, czy urządzenia dostępne na stanowisku laboratoryjnym są w stanie kompletnym, nie wskazującym na fizyczne uszkodzenie.
- Jeżeli istnieje taka możliwość, należy dostosować warunki stanowiska do własnych potrzeb, ze względu na ergonomię. Monitor komputera ustawić w sposób zapewniający stałą i wygodną obserwację dla wszystkich członków zespołu.
- Sprawdzić prawidłowość połączeń urządzeń.
- Załączenie komputera może nastąpić po wyrażeniu zgody przez prowadzącego.
- W trakcie pracy z komputerem zabronione jest spożywanie posiłków i picie napojów.
- W przypadku zakończenia pracy należy zakończyć sesję przez wydanie polecenia wylogowania. Zamknięcie systemu operacyjnego może się odbywać tylko na wyraźne polecenie prowadzącego.
- Zabronione jest dokonywanie jakichkolwiek przełączeń oraz wymiana elementów składowych stanowiska.
- Zabroniona jest zmiana konfiguracji komputera, w tym systemu operacyjnego i programów użytkowych, która nie wynika z programu zajęć i nie jest wykonywana w porozumieniu z prowadzącym zajęcia.
- W przypadku zaniku napięcia zasilającego należy niezwłocznie wyłączyć wszystkie urządzenia.
- Stwierdzone wszelkie braki w wyposażeniu stanowiska oraz nieprawidłowości w funkcjonowaniu sprzętu należy przekazywać prowadzącemu zajęcia.
- Zabrania się samodzielnego włączania, manipulowania i korzystania z urządzeń nie należących do danego ćwiczenia.
- W przypadku wystąpienia porażenia prądem elektrycznym należy niezwłocznie wyłączyć zasilanie stanowiska. Przed odłączeniem napięcia nie dotykać porażonego.